



Consultas Mais Complexas

CONSULTAS ENVOLVENDO MAIS DE DUAS TABELAS

Consultas envolvendo três ou mais tabelas funcionam da mesma forma que consultas com duas tabelas. O resultado da consulta consiste na seleção de linhas do produto cartesiano de todas as tabelas envolvidas.

Para uma consulta envolvendo 3 tabelas com 1.000, 5.000 e 10.000 linhas cada, o produto cartesiano origina uma tabela com 500 **bilhões** de linhas! Certamente os SGBDS não fazem o produto cartesiano antes da seleção das linhas. Eles utilizam algoritmos bastante sofisticados para simultaneamente combinar e selecionar de linhas. De qualquer forma, o desempenho de uma junção depende da quantidade de tabelas envolvidas, do tamanho destas tabelas e das condições impostas nas cláusulas WHERE e ON. Considere o diagrama ER a seguir:



As entidades FORNECEDOR, PECA e PRODUTO e os relacionamentos FORNECE e UTILIZA são mapeados em 5 tabelas. Os atributos não foram mostrados. Os comandos de criação e inserção de registros estão no material de apoio.

FORNECEDOR

id_forn	nome
1	Multi Peças
2	SP2 Peças
3	MM Peças
4	Dinâmica
5	Pégasus

PRODUTO

id_prod	nome
1	Motor CC
2	Motor AC
3	Eixo Transm.
4	Redutor

PECA

id_peca	nome
1	Mancal
2	Rolamento
3	Estator
4	Óleo
5	Engren. 1
6	Engren. 2
7	Engren. 3
8	Parafuso
9	Porca

UTILIZA

id_prod	id_peca
1	1
1	2
1	3
2	1
2	2
2	3
3	5
3	7
3	8
3	9
4	6
4	7
4	4

FORNECE

id_for[illegible]	[illegible]d_peca
[illegible]	1
[illegible]	2
1	3
2	1
2	2
2	3
2	4
3	2
3	4
4	4

A relação de **todos** os fornecedores e peças que fornecem:

```
SELECT F.nome AS FORNEC, P.nome AS PEÇA
  FROM FORNECEDOR AS F LEFT OUTER JOIN FORNECE AS R ON F.fornec_id = R. fornec_id
  LEFT OUTER JOIN PECA AS P ON P.peca_id = R.peca_id;
```

FORNEC	PEÇA
Multi Peças	Mancal
Multi Peças	Rolamento
Multi Peças	Estator
Super Peças	Mancal
Super Peças	Rolamento
Super Peças	Estator
Super Peças	Óleo
Mundo Peças	Rolamento
Mundo Peças	Óleo
Dinâmica	Óleo
Pégasus	NULL

Observe que foram feitos dois LEFT OUTER JOIN em sequência. Primeiramente, o mais à esquerda é executado e seu resultado é colocado em uma tabela temporária (virtual). Em seguida, o outro é executado entre a tabela temporária e PECA. Pode-se colocar em sequência quantas junções forem necessárias. A execução sempre será da esquerda para a direita e os resultados intermediários são colocados em tabelas temporárias. Na realidade, a consulta acima pode ser feita com um RIGHT OUTER JOIN e um INNER JOIN (por quê? verifique!):

```
SELECT F.nome AS FORNEC, P.nome AS PEÇA
FROM PECA AS P INNER JOIN FORNECE AS R ON P.peca_id = R.peca_id
   RIGHT OUTER JOIN FORNECEDOR AS F ON F.fornec_id = R.fornec_id;
```

O OPERADOR UNION

As operações de junção realizam sempre a seleção de linhas a partir do produto cartesiano das tabelas envolvidas. Estas tabelas possuem, na grande maioria das situações, estruturas distintas. O operador UNION permite que os resultados de duas ou mais consultas (SELECT) sejam unidas em uma só tabela. Algumas restrições: o número de colunas de todas as consultas envolvidas deve ser o mesmo; os tipos das colunas correspondentes em todas as consultas devem ser compatíveis.

```
SELECT ...
UNION [ALL | DISTINCT] SELECT ...
[UNION [ALL | DISTINCT] SELECT ...];
```

UNION retira os registros duplicados por padrão, sem que seja necessário incluir o modificador DISTINCT. Para forçar a inclu dos registros duplicados, o qualificador ALL deve ser usado. Um cuidado a ser tomado: o primeiro UNION [DISTINCT] que surgir, faz com que todos os registros duplicados até aquele momento sejam excluídos. Considere as tabelas a seguir:

A

a1	a2
1	7
2	8
3	9
3	9
5	11
6	12

B

b1	b2	b3
10	100	1000
20	5	11
30	300	3000
40	6	12

C

c1	c2	c3	c4
150	250	350	450
160	260	300	3000
170	270	6	12
180	280	380	480
190	290	390	490

```
(1) SELECT a1, a2 FROM A UNION ALL SELECT b2, b3 FROM B UNION ALL SELECT c3, c4
      FROM C

(2) SELECT a1, a2 FROM A UNION SELECT b2, b3 FROM B UNION ALL SELECT c3, c4 FROM C

(3) SELECT a1, a2 FROM A UNION ALL SELECT b2, b3 FROM B UNION SELECT c3, c4 FROM C
```

(1)

a1	a2
1	7
2	8
3	9
3	9
5	11
6	12
100	1000
5	11
300	3000
6	12
350	450
300	3000
6	12
380	480
390	490

(2)

a1	a2
1	7
2	8
3	9
5	11
6	12
100	1000
300	3000
350	450
300	3000
6	12
380	480
390	490

(3)

a1	a2
1	7
2	8
3	9
5	11
6	12
100	1000
300	3000
350	450
380	480
390	490



O OPERADOR INTERSECT

O operador INTERSECT faz o "oposto" do operador UNION: retorna as linhas que são comuns às consultas envolvidas. Embora alguns dialetos suportem o modificador ALL, o seu comportamento pode variar. Por conta disto, o funcionamento do INTERSECT será apresentado apenas com o modificador DISTINCT, que é o seu padrão.

```
SELECT ...
INTERSECT [ALL | DISTINCT] SELECT ...
[INTERSECT [ALL | DISTINCT] SELECT ...];
```

Considere as mesmas três tabelas A, B e C mostradas anteriormente.

```
SELECT a1, a2 FROM A INTERSECT SELECT b2, b3 FROM B INTERSECT SELECT c3, c4 FROM C;
```

a1	a2
6	12

```
SELECT a1, a2 FROM A INTERSECT SELECT b2, b3 FROM B;
```

a1	a2
5	11
6	12

Quando há mais de um INTERSECT na consulta, eles são executados da esquerda para a direita: A INTERSECT B INTERSECT C é equivalente a (A INERSECT B) INTERSECT C; para alterar a ordem de execução, devem ser usados parênteses.

Embora o operador INTERSECT esteja definido na linguagem SQL padrão, ele pode ser substituído por um comando INNER JOIN com as colunas correspondentes sendo comparadas na cláusula ON:

```
SELECT a1, a2 FROM A INNER JOIN B ON (a1 = b2 AND a2 = b3) INNER JOIN C ON (a1 = c3
AND a2 = c4);
```

a1	a2
6	12

O OPERADOR EXCEPT

O operador EXCEPT[2] ou MINUS seleciona as linhas da primeira tabela que não pertencem à segunda. Os modificadores ALL e DISTINCT têm a mesma função que no operador UNION.

```
SELECT ...
  EXCEPT [ALL | DISTINCT] SELECT ...
  [EXCEPT [ALL | DISTINCT] SELECT ...];
```

Considere as mesmas três tabelas A, B e C mostradas anteriormente.

```
SELECT a1, a2 FROM A EXCEPT SELECT b2, b3 FROM B EXCEPT SELECT c3, c4 FROM C;
```



a1	a2
1	7
2	8
3	9

```
SELECT a1, a2 FROM A EXCEPT SELECT b2, b3 FROM B;
```

a1	a2
1	7
2	8
3	9

Quando há mais de um EXCEPT na consulta, eles são executados da esquerda para a direita. Para alterar a ordem de execução, devem ser usados parênteses.

De forma similar ao operador INTERSECT, é possível implementar o operador EXCEPT utilizando o comando LEFT OUTER JOIN e um pequeno macete.

Deseja-se selecionar as linhas da primeira tabela (A) que não têm correspondente na segunda (B). Isto é o mesmo que excluir da primeira tabela as linhas que possuem correspondente, isto é, a interseção entre as duas tabelas. Deve-se incluir no SELECT um campo qualquer da segunda tabela. Este campo será NULL apenas nas linhas em que não há correspondência na segunda tabela. Deve-se também incluir o modificador DISTINCT no SELECT (o LEFT OUTER JOIN inclui todas as linhas de A, inclusive as repetidas):

```
SELECT DISTINCT a1, a2, b1 FROM A LEFT OUTER JOIN B ON (a1 = b2 and a2 = b3) WHERE b3
IS NULL;
```

a1	a2	b1
1	7	NULL
2	8	NULL
3	9	NULL

A precedência dos operadores UNION e EXCEPT é a mesma. Havendo UNION e EXCEPT na mesma consulta, eles são executados da e[...]erda para a direita. A ordem de execução pode ser alterada usando-se parênteses.

O operador INTERSECT tem prioridade sobre UNION e EXCEPT. Considere a consulta a seguir:

```
SELECT a1, a2 FROM A UNION SELECT b2, b3 FROM B INTERSECT SELECT c3, c4 FROM C;
```

O operador INTERSECT será executado primeiro

```
SELECT b2, b3 FROM B INTERSECT SELECT c3, c4 FROM C
```

produzindo o resultado abaixo:

b2	b3
300	3000
6	12

Em seguida, a operação envolvendo UNION é executada

```
SELECT a1, a2 FROM A UNION {resultado}
```

A

a1	a2
1	7
2	8
3	9
5	11
6	12
300	3000

[1] O MySQL não implementa o operador INTERSECT em seu dialeto SQL.

[2] O MySQL não implementa o operador EXCEPT em seu dialeto SQL.

Atividade Extra

O padrão SQL:99 incluiu a cláusula WITH RECURSIVE, permitindo o uso de consultas com recursividade. Por exemplo, considere a seguinte tabela (simplificada):

```
CREATE TABLE funcionario (
    id_funcionário INT(6) NOT NULL PK,
    nome VARCHAR(45),
    id_chefe INT(6) NOT NULL,
);
```

O campo *id_chefe* aponta para o registro do chefe deste funcionário (autorrelacionamento). É possível relacionar todos os funcionários que são também chefes e os funcionários de um determinado chefe. Porém, não é possível percorrer a hierarquia de chefia (chefe do chefe, chefe do chefe do chefe etc.) sem o uso de recursividade.

Pesquise o uso deste recurso e implemente uma consulta que percorre recursivamente a tabela *funcionário* mostrando toda a hierarquia de chefia.

Referência Bibliográfica

- ELMASRI, R. e NAVATHE, S. B. **Sistemas de Banco de Dados**. 7ª Ed., São Paulo: Pearson, 2011.

- RAMAKRISHNAN, R., GEHRKE, J. **Sistemas de Gerenciamento de Banco de Dados.** 3ª Ed., São Paulo: McGraw-Hill, 2008.

- CORONEL, C. e ROB, P. **Sistemas de Banco de Dados - Projeto, Implementação e Gerenciamento.** 1ª Ed., São Paulo: Cengage, 2010.

- GROFF, J. R., WEINBERG, P. N. e OPPEL, A. J. **SQL: The Complete Reference.** 3ª Ed., Nova York: McGraw-Hill, 2009.

Ir para exercício